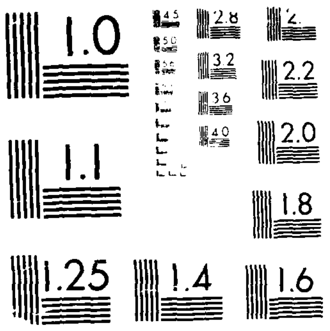
AD-A192 992   MANAGING MICROCOMPUTER APPLICATIONS: A PRIMER AND GUIDE   1/1
TO GOOD PRACTICE(U) ARMY ENGINEER INST FOR WATER
RESOURCES FORT BELVOIR VA R M MALES ET AL. MAR 88
UNCLASSIFIED   IWR-88-R-3   F/G 12/6   NL



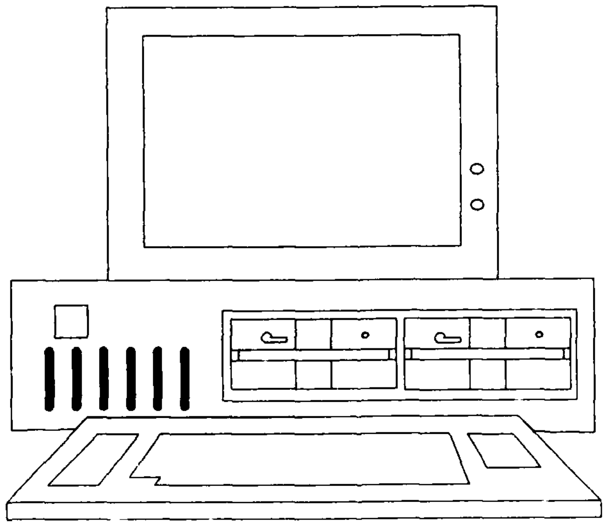
MICROCOPY RESOLUTION TEST CHART

[illegible]

4

US Army Corps of Engineers
Water Resources Support Center
Institute for Water Resources

Managing Microcomputer Applications: A Primer and Guide to Good Practice



March 1988

IWR Report 88-R-3

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188 Exp Date Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): IWR Report 88-R-3	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Water Resources Support Center Institute for Water Resources — 6b. OFFICE SYMBOL (If applicable): CEWRC-IWR	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Casey Building Ft. Belvoir, VA 22060-5586	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: U.S. Army Corps of Engineers — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): 20 Mass. Ave. Washington, DC 20314-1000	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Managing Microcomputer Applications: A Primer and Guide to Good Practice

12. PERSONAL AUTHOR(S)
Richard M. Males and Michael R. Walsh

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM ______ TO ______	88/3/31	88

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Managing Microcomputers, Organizational & User Needs, Potential Problems with Microcomputers, End-User Computing & Applications, Types of Software

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
This report has a dual purpose. First, the report is a primer for managers to help them understand the potential impacts of the microcomputer on organization and staff productivity. The most common types of microcomputer software are described and basic types of applications developed by planners are discussed. Second, the report is a guide for managers faced with managing the use of microcomputers and the development of applications by their staff and others. A process for managing the development of "corporate" applications is presented.

The report is directed at a "non-computer professional" audience, i.e. managers within the Corps who have a technical background, but may not be microcomputer users themselves, and have as part of their responsibility the management of individuals and/or projects in which microcomputers are used. The report is designed primarily to raise awareness of the need for, and the methods of, management of microcomputer applications. Outline formats are often used, and key ideas are highlighted. It is hoped that this format will communicate the key concepts better than a more traditional report.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Michael R. Walsh	22b. TELEPHONE (Include Area Code): (202) 355-3087 — 22c. OFFICE SYMBOL: CEWRC-IWR-R

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

4

Managing Microcomputer Applications:

A Primer and Guide to Good Practice

by

Richard M. Males

and

Michael R. Walsh

U.S. Army Corps of Engineers

Water Resources Support Center

Institute for Water Resources

Fort Belvoir, VA 22060-5586

March 1988

IWR Report 88-R-3

Table of Contents

Introduction

Purpose of This Report

Microcomputers are becoming common tools within Corps planning offices. However, microcomputers do not appear to be managed as are other organizational resources. Management often focuses on acquisition and standardization of hardware and software, but the actual usage of the microcomputer by staff, in particular to develop applications, such as spreadsheets and databases, is seldom managed. Microcomputer users are often left to "do their own thing", with scant management attention to the impacts on the organization as a whole in terms of efficiency, consistency, productivity, and security.

This report is a primer on microcomputer use and a guide for developing microcomputer applications.

The purpose of this report is twofold. First, the report is a primer for managers to help them understand the potential impacts of the microcomputer on organization and staff productivity. The most common types of microcomputer software are described and basic types of applications developed by planners are discussed. Second, the report is a guide for managers faced with managing the use of microcomputers and the development of applications by their staff and others.

This report is directed at people who manage microcomputer users in the Corps.

The report is directed at a 'non-computer professional' audience, i.e. managers within the Corps who have a technical background, but may not be microcomputer users themselves, and have as part of their responsibility the management of individuals and/or projects in which microcomputers are used. The report is designed primarily to raise awareness of the need for, and the methods of, management of microcomputer applications. Outline formats are often used, and key ideas are highlighted. It is hoped that this format will communicate the key concepts better than a more traditional report.

Background

In 1985, the US Army Engineer Institute for Water Resources (IWR) performed a study entitled "Needs Assessment of Corps Planning Information Management Systems". This effort was directed at exploring the methods by which planning managers within the Corps

Microcomputer use within the Corps has increased significantly, particularly for 'local' information systems, but:

- *there is a need for training;*
- *many systems are not documented;*
- *there is little information transfer.*

used microcomputers for management of planning studies. The findings were documented in IWR Contract Report No. 85-C-5 (August 1985). The study showed the **significantly increased use of microcomputers in the Corps for 'local' information systems**, but also **pointed out important problems in terms of needs for training, lack of information transfer within the Corps on such systems, and lack of documentation and use of good design practice** in the development of such systems. As an outgrowth and 'follow-on' to the previous study, the current study, "A Process for Managing Corps Planning Information" was carried out by IWR in 1986-1987. This work was directed towards enhancing information transfer within the Corps, through development of an 'applications catalog' of Corps-developed planning management microcomputer applications, and towards improving the management of microcomputer resources, in particular in terms of developing and maintaining microcomputer applications. This report is one of a series of products of the current study, directed at the issue of management of microcomputer applications. An additional product, a report entitled "Microcomputer Applications in Planning Catalog", (IWR Contract Report No. 87-R-9) consisting of program abstracts relating to Corps-developed microcomputer applications, is also available from IWR.

Development of this Report

Many microcomputer applications are abandoned after considerable expenditure of time and effort.

As noted above, the need for management of microcomputer applications within the Corps became apparent in a prior effort. As a part of the current study, many of the microcomputer applications that had been identified previously were re-examined - in a significant number of cases, these projects had been abandoned after major investments of time and effort. In addition, during the course of the past two years, data processing and computation activities within the Corps have undergone a reorganization with the creation of the Directorate of Information Management. At the same time, microcomputer equipment became more widely available throughout the Corps.

In light of these changes, it was felt that a report directed primarily at managers, highlighting some of the issues and problems associated with obtaining good productivity from microcomputer usage, would be appropriate. Managers within the Corps, when attempting to manage microcomputer applications, are presented with some problems that are unique to the Corps, and some that are general in character. Development of this report was based on review of the emerging literature on management of computer usage and on interviews and discussions with microcomputer specialists and managers inside and outside the Corps who have worked on and been concerned with this issue.

Acknowledgements

This study was coordinated by Mr. Michael Walsh of IWR, and carried out by Dr. Richard M. Males, RMM Technical Services, Inc., 3319 Eastside Avenue, Cincinnati, Ohio 45208, as sub-contractor to Planning & Management Consultants, 808 West Main Street, Carbondale, Illinois 62901.

This report was prepared using the Xerox Ventura Publisher (tm) desktop publishing program, and printed using an HP LaserJet II printer. Mention of specific software within the report does not constitute an endorsement of the product, rather it is recognition of a particular place that product has achieved in the world of microcomputer usage, such as Lotus 1-2-3 or dBase III as 'de facto' standards for spreadsheet and data base software respectively, through virtue of market share. Lotus 1-2-3 and Symphony are trademarks of Lotus Development Corp. dBase III and Framework are trademarks of Ashton-Tate, and SuperCalc is a trademark of Computer Associates.

What Does a Manager Need to Know?

Microcomputers have become a fact of life, but many managers are not equipped, by training or inclination, to deal with 'computers everywhere'.

A manager can easily be at a loss as to how to best deal with microcomputer usage under his/her control. A manager needs to know certain basic concepts about microcomputers and microcomputer users. **It is not necessary to become an expert, but, as with any other arena of management, you need to know enough to know what questions to ask, and how to evaluate the answers.**

A manager must be aware of certain general principles and methods of organizing microcomputer use; must be cognizant of the overall climate and approach to computer usage within the manager's organization; must have some degree of knowledge of the technical concepts and tools involved; must be aware of, and sensitive to, personnel issues; and must have a grasp of the techniques necessary to insure productive use of microcomputers. The following is an outline of some of these concepts; the remainder of this report will treat these issues at greater length.

Basic Organizational Concepts

How do we start thinking about managing microcomputers?

- Organizational Requirements and User Needs
- End-User Computing
- The Application as an Organizing Unit

Basic Technical Concepts

What are some key technical concepts?

- Types of Software
- Generalized Applications Packages
  - Word Processing
  - Spreadsheet Packages
  - Data Base Management Packages
  - Communications, Uploading, and Downloading

Personnel Issues

What about people?

- Motivating factors, characteristics, and styles of 'computer types'
- Skill levels

The Applications Development Process

How should applications be developed?

- Requirements analysis, design documents, design review
- Programming/coding/testing
- Installation
- Performance review
- Maintenance - archiving

Documentation

- Purposes and levels of documentation

Quality Control Techniques

- Design and review methodologies

Advanced Tools for Applications Development

- Productivity tools
  - Multi-tasking
  - Auditing/Computer-based documentation aids
- Structured design techniques

Microcomputer Management in the Corps

Managers of microcomputer users must contend with the organizational and attitudinal realities within the Corps.

Microcomputer management within the Corps must take place within the context of overall Corps organizational issues. The Corps has recently re-organized to form the Directorate of Information Management (DIM), and to form Information Management Offices (IMO's) within each Field Operating Agency (FOA). While the IMO functions encompass management of microcomputers, the initial thrust is towards the traditional, large machine systems of the Corps. With regard to microcomputers, IMO has emphasized coordination of hardware/software acquisition through the purchase contract with Zenith Corp., and programs for training and support of microcomputers are not as yet widespread within the Corps. Consequently, **the Corps manager with responsibility for individuals who use microcomputers must, at present, rely on his/her own skills and initiative to insure appropriate and effective usage of these systems.**

Organizational Issues

Organizationally, the Corps is just beginning to provide strong support for microcomputer users and managers.

ADP/IMO has traditionally been separated organizationally from the users. Large time demands on ADP/IMO for administrative support and programming have made it difficult for ADP/IMO to respond effectively to support 'end-user computing'.

The overall Corps approaches to information systems are focused on centralized, not de-centralized (i.e. emphasizing microcomputers) systems..

Corps Planning Managers are ultimately on their own as to how effectively microcomputers are used in their planning offices.

Personnel mobility within the Corps creates problems, as applications developers, who may be the only ones with detailed knowledge of applications, move from one group to another or leave the Corps. There is no career ladder for engineers or other technical individuals within the Corps who wish to keep their technical roles, but concentrate on building their skills in computer usage, and applying these skills in the planning tasks at hand.

There are few formal support functions for microcomputer users: it is difficult to assign a designated microcomputer support person within a work group, with time allocated for this function. In industry, the 'microcomputer resource center' is increasingly common, where users can receive training, try out different software and hardware packages, etc., but is not widespread in the Corps.

Attitudes

Computer usage is often looked on as an 'extra', not a person's 'real' job.

Communications problems with ADP groups in many FOA's have created a poor climate for obtaining assistance in the development and use of applications. ADP groups are frequently not oriented towards microcomputers. Many managers have an 'anti-computer' bias, in particular relating to productivity uses such as word processing. Computer applications are frequently developed by planners while continuing to accomplish assigned work.

The large machine systems of the Corps have not worked well to provide information to the managerial levels at FOA's, and are looked upon by managers as a burden of upward reporting. This has produced a negative attitude towards many centralized computer systems, and sometimes to computers in general.

Training

Many individuals using computers within the Corps are self-taught. Managers may not understand computer issues or recognize the need for managing the computer usage under their authority. Training in issues of management of computer resources is not provided.

Managing Microcomputers - Organizational and User Needs

All too often, microcomputers are a mixed blessing in an organization.

Microcomputers can clearly provide great increases in productivity and new opportunities for problem-solving within an organization. **The introduction and use of microcomputers, however, is not without problems:**

- Many individuals who would potentially benefit from using microcomputers require significant encouragement, training, and 'hand-holding'. Training costs, and a long learning curve before productivity becomes apparent, are real concerns.
- The organization itself must re-organize to provide the needed support, and be willing to bear the costs of training, support, and experimentation.
- Excessive enthusiasm can be a problem. Significant resources may be devoted to the 'computer' aspects of the job, at the expense of the 'real work' to be done, or tasks may be computerized when they should not be.
- Reliance on a single knowledgeable individual to handle computer-related tasks can place the organization in a vulnerable position.
- Large investments may be made in systems that are inadequate, difficult to maintain, and rapidly become obsolete. Organizational concerns for security, productivity, and some level of standardization may be ignored.

Organizational needs for productivity, security, and standards are inherently in conflict with user desires for immediacy and flexibility of use.

The highly individualized nature of microcomputers is frequently at odds with organizational needs and norms. Unlike large computers, the microcomputer is inherently and technically free of controls. A user is limited only by the hardware/software capabilities, and his/her own skill. The microcomputer user needs encouragement, support, training, and time to experiment. The microcomputer user does not want controls, or 'organizational standards', to intrude on use of the computer. The user values the immediacy of the microcomputer, the quick response, and the direct access to computational capabilities.

Organizational needs are broader, and more long-term, than user needs. The organization needs to insure that the resources are used properly in pursuit of its goals. This requires controls, monitoring, and standards, all things that get in the way of the user's desire for immediacy. Controls, monitoring, and standards bring back many of the problems that encouraged the migration away from large computers to small computers in the first place. The problem is to create and put in place **the minimum level of controls needed to insure that the organization's needs are met, without stifling the individual creativity, productivity, and ease of use that microcomputers make possible**, and, at the same time, providing needed support and encouragement.

How can a manager best handle the issues of microcomputer usage, gaining the advantages while minimizing the problems?:

Microcomputers will not manage themselves. In the absence of active management, 'messes' will be created.

- first, the manager must **accept the need to manage** the microcomputer resource, just as any other resource in an organization is managed;
- next, the manager must **learn and understand the issues and concepts** important to microcomputer management;
- finally, the manager must **adopt and use** the necessary *management practices* to insure quality, appropriateness, and security of microcomputer applications.

Computer Issues

Mainframes -

Problems of:

- *Long Turnaround*
- *Complex Access*
- *Not 'friendly'*
- *Need ADP Professionals*
- *Hard to Learn*
- *Not Always Available*
- *Too Many Procedural Controls*

The advantages of the microcomputer have highlighted the problems associated with using large computers. More and more frequently, the perceived solution to problems of getting computer support from large computers is to 'do it yourself with a microcomputer'. This solution is not without its own problems.

Large machine usage is primarily through the assistance of data processing 'professionals'. Microcomputer usage is primarily through 'end-user' computing, i.e. by individuals who are not data processing professionals. This brings both advantages and disadvantages.

While microcomputers have made undeniable contributions to individual and organizational productivity, they are not without problems:

- When all the costs are added in, they are not necessarily a cheap resource.
- They may change the way people do business, not necessarily for the better. Problems may be forced into molds and redefined so that the microcomputer can solve them.
- Individuals may prefer working on microcomputers to doing their 'end product' work, or may spend excessive amounts of time in using them.
- Individual choices as to how to do things may be at odds with choices of other individuals in the same group, or with organizational norms, leading to problems and incompatibilities later on.
- In an environment in which microcomputers are shared people may end up waiting on the availability of the microcomputer.
- Microcomputer users may be self-taught, or minimally trained. There may be much 're-invention of the wheel', as each user goes painfully through the same learning curve.

Desktop Computers -

Problems of:

- *Logistics and Resources*
- *Training and Skills*
- *Attitudes*
- *Organizational Issues*

Managerial or individual attitudes towards microcomputers may result in problems, either through over-enthusiasm for what can be accomplished by a microcomputer, through lack of appreciation for the productivity possibilities, or through fear and lack of knowledge. Knowledge and skills may be 'hoarded' by individuals, as a source of organizational power.

Lack of standards, controls, policies and procedures, and inadequate support mechanisms, can create a situation of incompatibility, vulnerability, and lack of justification and documentation within the organization.

Managers must recognize that the current differentiation between use of mainframes or minicomputers through terminals, and use of standalone microcomputers, will diminish over time. Microcomputers will be increasingly interconnected to other microcomputers in networks, or act as workstations and terminals for mainframes and minicomputers. Thus, the 'wave of the future' is increasingly powerful microcomputer workstations, interconnected amongst themselves and to other computers, with sharing of data and programs, simpler access to 'corporate' data bases on mainframes, and increased and simpler use of communication facilities such as sending electronic messages and mail. Over time, the currently wide distinction between use of 'personal' and centralized computing facilities should diminish. This will, not, unfortunately, diminish the need for management.

Potential Problems With Microcomputers

Logistics And Resources

- *Availability*
- *Processing Power*
- *Output Limitations*

Usage may be shared between individuals or a group. If the microcomputer is not immediately at hand, it is less likely to be used. A microcomputer may not be sufficiently powerful for the problem to be solved, resulting in long processing times, or requiring that problems be subdivided. Speed of printing may be a limiting factor. The environment in which the microcomputer is used can also be a problem. Proper lighting, ergonomic furniture, and sound-proofing are required to make using a microcomputer comfortable and less stressful for users. These considerations are often omitted with resulting complaints by users.

Training And Skills

- *Long Learning Curve*
- *Diverse Learning Styles*
- *Needs for Hand-Holding and Support*
- *Varied Skill Levels*
- *Appreciation of Good Design*

Some individuals may be overwhelmed by the great deal of information that has to be assimilated to use a microcomputer effectively. There is a long learning curve. Acquiring the necessary skills may take a long time before productivity increases are seen. There are significant differences in learning styles: some users respond to 'learn-by-doing', some to classroom training, some to learning from books. There is a general need for 'hand-holding' and skilled assistance. Ready accessibility of support by more skilled users is generally required by less-skilled or beginning users. This can create excessive demands on the skilled users in an organization. There is generally little knowledge of good design practices. Many users are self-taught. They concentrate mainly on getting the job done, and learn by doing. Good design techniques are appreciated and acquired much later, if at all. Software choices are often poorly made. Often, the choice is not necessarily what is best for job, but what is available, familiar, or looks like it would be 'fun' to learn.

Attitudes

- *Computers not viewed as personal productivity tool*
- *Personalized Resource*

Microcomputers may not be viewed as a productivity tool by management. Comments such as "engineers don't type" or "this takes longer to get out of the computer than doing it by hand" reflect this attitude. A productivity tool is one that enables planning staff to improve job performance. Properly used, a microcomputer is a productivity tool.

Microcomputers tend to become a 'personalized' resource, not an organizational resource. Microcomputers are not seen as a critical resource to be managed - managers often will assume the best, ignore the problem of managing microcomputer usage, or delegate it and forget it.

Organizational

- *Computer usage not part of regular job description*
- *Reliance on Key Individuals*
- *Need for a Support System*
- *Need for Procedures and Standards*

Development of computer applications is not seen as part of the basic job. Applications may be 'bootlegged', developed 'on the side' while still performing regular work. Conflicts between regular work and 'computer work' become inevitable. Frequently, single individuals are relied on to develop systems: the organization becomes vulnerable to the departure of key persons, skilled individuals become overworked, and the lack of a 'team' approach means that there is no critical review of systems by peers. Lack of a formal support system for users results in reliance on an informal support system: skilled users are relied upon, with attendant costs of their time, lack of progress when they are not available, and frustration on the part of both the more and less-skilled users. Time is consumed in seeking out information, and experimenting. Lack of procedures and standards can lead to poorly designed and undocumented computer efforts.

Getting on Top of the Situation (things to remember)

Managers must be sensitive to the signs of a problem with microcomputers, be aware of the available management tools, and look to promote attitude changes, in themselves and in the microcomputer users.

What are the Symptoms of a Problem?

Look for signs that management is needed.

- Unhappy Users
- Unhappy Managers Of Users
- The Manager Doesn't Really Understand What Is Going On
- The Manager Doesn't Really Want to Understand What is Going On
- Information From Systems Is Not Timely
- Information From Systems Is Not Used
- Computer Applications Are Hard To Change
- Too Much Time Is Spent Manipulating Information
- The Wrong People Are Manipulating Information
- Bad Report Formats
- Reports Not Available in a Timely Fashion
- Applications Are Discarded Shortly After Development
- Only A Few Individuals Understand The Computer Applications
- Computers Are Available, But Seldom Used

What are the Management Tools?

- Organizational Framework
- Policies
- Attitudes
- Procedures and Standards
- Training and Support Systems

What are the Attitude Changes?

Managers should:

- accept the need to manage microcomputer usage as part of their managerial job

- accept the microcomputer as a personal productivity tool (if not for themselves, then for others)
- recognize the need for organizational and individual learning curves before productivity benefits are realized
- search for the least intrusive, minimal controls needed
- accept the need for some organizational changes, and the institution of support systems for microcomputers

Successful management of microcomputer usage requires attitude changes on the part of both the manager and the microcomputer users.

Users should:

- accept that the microcomputer is an organizational, not individual resource
- accept that the organization has legitimate interests that may not match the user's immediate interests
- accept the need for some controls, standards, and policies to implement management interests
- recognize the need to improve their own practices, both for organizational and individual benefits
- accept the responsibility of teaching and supporting their colleagues with knowledge that they have obtained

End-user Computing and Applications

The concepts of 'end-user computing' and 'applications' are two important, related organizing concepts of microcomputer usage that a manager must understand.

End-user Computing

End-user computing is the most significant development in microcomputer use.

- Microcomputer usage by an individual, almost always someone who is not a trained computer professional
- Frequently through use of a flexible 'packaged' program, such as a data base management or spreadsheet program

Once users did not need to learn a programming language, access to computing no longer needed to be through 'professionals'.

End-user computing refers to the capability of each individual computer user to do productive work with a microcomputer, without the need to learn a programming language. This is made possible by the availability (on both large and small computers) of sophisticated programs that assist the user in such efforts. These programs are generally spreadsheet programs (such as Lotus 1-2-3 or Symphony) or data base management programs (such as dBase III).

Applications

The 'Application' is the basic unit that can and should be managed.

In discussing computer resources, the 'application' is a useful organizing concept. Rather than talking about hardware, software, documentation, or programs, **an application is the combination of these resources to solve a particular problem.** Thus, an application can consist of a single program, or of multiple programs. It may involve one computer, or many. Within a given organization, there may be multiple applications. **Microcomputers, coupled with powerful computer software such as data base management, word processing, or spreadsheet software, have made it very easy for applications to be developed.** This is one of the great advantages of the personal computer 'revolution', and one of the greatest pitfalls, as well.

Examples of such applications include:

- standard documents maintained in word processing format

- mailing lists maintained in a data base management system
- budget worksheets using a spreadsheet program

'Corporate' and 'Personalized' Applications

For purposes of management, a distinction can be made between **'personalized'** applications and **'corporate'** applications. Personalized applications are those which are short-term and single-user in nature, involving minimal use of resources. 'Corporate' applications, even if developed and used by a single individual, are those that have wider implications for the organization as a whole, either through the resources necessary to accomplish them, or the fact that interactions with others are required for design and use of the application.

Concentrate on managing computer usage by managing 'corporate' applications.

Every instance of personalized application development need not be 'managed' through procedures and controls, although good practice should be encouraged at all levels of effort. Users need some freedom to experiment and learn, and excessive control, particularly for small efforts, destroys the immediacy and value of microcomputers. 'Corporate' applications, that involve development efforts of more than one day, or that are developed by one individual for another, should be managed through an orderly process of justification, development, and review.

Determining the Degree of Problem

Questions a Manager Should Ask

A manager should periodically perform a self-assessment relating to the applications for which he/she is responsible. Such an assessment is an excellent first step in determining the degree of need for management, and in raising awareness of the issues.

Some questions that should be asked:

- What applications are being used by my staff?
- Do I know **WHY** they are being used?
  - to do something previously done in another fashion?
  - to do something new?
  - to do something that doesn't need to be done?
- Are there documents describing the applications?
  - at what level of detail do they describe the application?
  - where are they?
  - can I understand them?
- Do I know how much time was spent in development of each application?
  - Does it seem reasonable?
- Do I know how much time is spent in use of each application?
  - Does it seem reasonable?
- Does the value of each application appear commensurate with the effort associated with development and use?
- Does more than one individual understand each application?
- Has the development of each application been justified and approved?
- Does a written, intelligible, design document exist for each application?
- Has each application been tested and verified?
- What quality assurance procedures are in place?
  - for a spreadsheet, to insure that calculations are correct
  - for a data base, to insure that data is correct

- How much time does it take to change an application, if needed?
- Are changes needed often?
- How good is the backup of the application?
- Are we protected against loss?
- Are there privacy and/or security considerations?
- Will an application be usable if the original developer no longer works here?
- Do I have any procedures in place to prevent duplication of effort, or 're-inventing the wheel'?

Types of Software

A manager must be aware of the variety of different kinds of software that are available, because software is a major component of applications. The major types of software of concern are: operating system software; programming languages; packaged computer programs, and commercial applications packages.

Operating System Software

Operating system software is the overall controlling program for the microcomputer. The operating system software handles the 'housekeeping' of the computer, transferring information from disks to memory, and to the printer. The operating system software determines the 'user interface', i.e. the method by which the user communicates with the computer.

There are a number of incompatible operating systems used by microcomputers. The IBM (and compatible) microcomputers use a 'Disk Operating System' (DOS), in various versions, two being called PC-DOS or MS-DOS. DOS is a single-user operating system, that is, only one individual at a time can make use of the computer, in general to do only one thing at a time. Recent announcements of a new version of the operating system for advanced IBM and compatible microcomputers (OS/2) hold out the promise of **multi-tasking**, i.e. the ability to simultaneously perform more than one program within the microcomputer, but the operating system will still be single-user, i.e. one person at a time using the computer.

UNIX, and its various derivatives, such as XENIX, are multi-user operating systems, that allow many users to hook into a single microcomputer, allowing for sharing of information and data, as is common with minicomputers and large mainframes. For a variety of reasons, the Unix operating systems have not achieved much popularity outside of scientific and technical applications.

The other major competing operating system is the Macintosh operating system, used on Apple's Macintosh series of computers. This operating system is distinctly different from the text oriented DOS and Unix, which require the user to type in commands to instruct the computer. The Macintosh operates in a graphic format, using so-called 'icons', small symbols that appear on the screen to represent programs, data files, etc. The user selects the desired functions by 'pointing' with a 'mouse', a device that allows the users, by moving the mouse on a desk, to change the position of a pointer on the screen.

The graphically-oriented Macintosh user interface is significantly different from the text-oriented operating systems of Unix and DOS, and is considered by many to be easier to learn and to use. In addition, all programs that use the Macintosh operating system share this common user interface, while on the IBM series of computers, each program may require the user to interact differently.

Text and graphics-based systems are merging with new developments in operating systems. Forthcoming IBM-based operating systems are expected to use many of the graphical approaches common to the Macintosh, and many programs developed for the IBM PC currently have the 'look and feel' of Macintosh programs (particularly those emphasizing graphics or desktop publishing applications). The user gains from these developments, in terms of productivity, ease of use, and simplicity of learning.

In general, the choice of an operating system is implied by the choice of computer. Corps managers need to be aware of the issues, but will generally have technical support and direction in making these choices. Based on existing Army purchase contracts, the IBM operating system, and IBM-compatible PC's, are the primary systems used in most offices, with Macintosh systems less common, and often acquired for specific purposes (again, graphics and desktop publishing).

Programming Languages

Computer programs are written in languages, such as BASIC, PASCAL, COBOL, FORTRAN, and C. Software that allows the use of such languages to develop applications is called a 'compiler' or 'interpreter', depending upon the particular technology used. Until a few years ago, anyone wishing to operate a microcomputer had to become knowledgeable in a programming language, and learn the skills and techniques of programming, frequently a difficult task. Using a programming language, a programmer is essentially unrestricted in terms of what the computer can be made to do.

Packaged Computer Programs

Programs (written in a programming language) , designed to fulfill a specific function, such as accounting, or some engineering calculation, are widely available. The user is unable to change anything except the data upon which the program operates. Such programs can be purchased directly, or can be developed by a programmer (usually a costly task).

General Purpose Applications Programs

The general purpose applications program is a special type of program, one which presents the user with a defined set of capabilities, but at the same time allows the user to design, define, and execute various tasks, changing not only the data, but the method by which it is processed. These programs have grown up as it has been recognized that there are a number of common tasks that can benefit from computerization - text processing, handling data, graphical display, and 'spreadsheet' format calculations. The program provides a framework for handling certain types of information, but does not restrict the user - rather these programs are designed to simplify the user's tasks in developing his/her own applications. This type of program has become the most popular of all types of microcomputer program.

General purpose applications programs have placed the power of the microcomputer in many hands - no longer is it necessary to learn an arcane programming language to get the computer to do useful work.

While the use of computers no longer requires the programmer as an intermediary, the 'de-professionalization' of computer usage means that many individuals, without knowledge of the established techniques that have been developed over the years to make computer programs easier to develop, maintain, and use, are designing and developing computer applications of varying degrees of utility. **Managers must be aware that use of commercial applications programs such as spreadsheet or database management programs is a form of programming, and that there are methods and techniques in existence that are designed to promote better applications development.**

Text Editing/Word Processing

Word processing is one of the most common, powerful, and productive forms of microcomputer usage.

Of all of the general purpose applications package types, **word processing generally has the shortest 'learning curve'**, i.e. the time to doing productive work using word processing is the shortest. Word processing involves entering text information into a computer (generally by typing it in, although other methods, such as automatic scanning of printed documents, are becoming available) where it can be modified, stored, and printed in a desired format. 'Standard' documents that are used repetitively can be created, and easily customized as needed. Additional capabilities associated with word processing include checking a document for spelling, providing a thesaurus, and checking for grammar, usage, and readability. 'Mail merge' capabilities combine a data base with a 'fill in the blanks' document to create letters that can be 'personalized' to each member of a group. It is also possible for many members of a work group to edit and annotate a single document electronically, just as a draft document is normally passed around among individuals for review and comment.

Word processing is often looked upon as a 'secretarial' task, but managers should recognize that anyone who generates documents can, if desired, make profitable use of word processing. Within an organization, use of compatible word processing between secretarial and technical/professional individuals allows for a variety of options - individuals can develop a document in draft form, that can be formatted and completed by secretaries, and resubmitted to the author, for final editing and correction, all electronically, and then returned to the secretary for printing.

Unlike the data base and spreadsheet types of commercial applications package, there is no single 'industry standard' word processing software that has become, through market share, the dominant 'player'. Different software packages have different capabilities, and emphasize different arenas - ease of use or learning, capability for complex text formats, speed, etc. While the selection is often made as a personal choice or out of habit ('I've always used this program'), managers should look to insure consistency within a work group, or, at minimum, that document formats can readily be translated between all the word processors used within the group.

Managers should be aware that so-called 'dedicated' word processing programs are in general much more powerful and easier to use than the word processing components of 'integrated' software that exists primarily for another purpose (e.g. spreadsheet or data base management).

A recent advance in the arena of text processing is the popularity of **desktop publishing**. Desktop publishing involves the use of laser printers (now available for under $2000) and specialized text processing software that allows for generation of different type faces and styles, layouts, and the inclusion of graphics and charts within documents. [This document is an example of desktop publishing]. Desktop publishing software is at the early stages of its development, and currently requires a fairly long learning curve. As with other forms of software, the flexibility and power of this capability can be abused, leading individuals to spend more time on the format of documents than on the content, and to worry about 'prettiness' when perfectly adequate communication can be accomplished by simpler means. However, the ability to communicate effectively on the printed page, merging text and graphics from a variety of sources, is a strong reason for examining desktop publishing capabilities. Over time, it is likely that desktop publishing capabilities will be increasingly included in 'standard' word processing packages.

Spreadsheet Programs

A spreadsheet program is a computer analog of the manual 'spreadsheet' for numerical calculations on data. Data entry, editing, and most importantly, recalculation, are done simply.

The spreadsheet is a very popular form of microcomputer software. Spreadsheet programs initiated the popularity of microcomputers in the business world (Visicalc on the Apple II,1979). With the introduction of Lotus 1-2-3 (1983) for the IBM PC, spreadsheet software (and Lotus as the de facto standard) became very widely used. Lotus 1-2-3, Symphony, SuperCalc, and Framework are the most popular packages with spreadsheet capability. So-called 'integrated' packages (such as Framework and Enable) add features beyond simple spreadsheet manipulation, such as graphics, word processing, data base management, and communications. In general, these additional capabilities are not as powerful as those available in stand-alone wordprocessing, data base management, communications, or graphics packages, but do provide a basic level of functionality.

Spreadsheets organize information in cells within a matrix. Each cell is given an 'address', e.g. cell A1, cell B3, to designate its row-column position in the matrix.

- In general, in each cell, one of three types of information can be stored:

a number	1.345
a 'label' (text)	Revenues
a formula	+A1*3.5

Formulas are used to calculate information based on values stored in other cells, and provide the great power of spreadsheets. In the above example, the formula says 'multiply the value in cell A1 by 3.5'. This value is then assigned to the cell in which the formula is located, and can be used for any further calculations. As numbers in cells are changed, automatic recalculation of all formulas takes place. Thus, calculation problems such as preparation of budget estimates or workload projections become vastly less tedious, as the spreadsheet handles all of the numerical calculation efforts automatically.

The array of numbers, labels, and formulas constitutes a basic spreadsheet. Additional capabilities are provided by 'macros', a set of instructions that can be stored within the spreadsheet itself to carry out more complex operations, or make things easier for the user.

The macro 'language' can be very difficult to understand. Below is a sample of a macro, taken from a popular book on Lotus 1-2-3 [Andersen]. This macro is used to change a row of labels to numbers. Clearly, the meaning of these instructions is not self-evident:

Spreadsheets have their own 'programming language'. A 'Macro' is an instruction written in the programming language of a spreadsheet.

```
\N      /rncHERE~~
        /xi@isna(HERE)~/xgEND~
        {edit}{home}{del}~
        /rndHERE~~
        /xg\N~
END     /rndHERE~
        /xq~
```

Spreadsheet programs provide great computational power in a relatively simple to use form. Managers in particular can make good use of spreadsheet capabilities for planning, budgeting, tracking, workload projection, and cost estimating. Spreadsheets capabilities, however, can easily be used inappropriately. **A manager must be sufficiently familiar with spreadsheets to know when the spreadsheet format is the apprcpriate choice for an application** (as opposed to data base, word processing, or programming language approaches).

Advantages of Spreadsheets

- It is easy to create simple spreadsheets, once the basic skills are acquired.
- The spreadsheet format is applicable to wide variety of problems.
- There is a natural, 'intuitive' structure and familiar format of results.
- It is easy to change numbers and assumptions. One of the great strengths of spreadsheets is the ability to do 'what if?' calculations, i.e. rapidly evaluating a number of alternatives.
- Spreadsheets operate quickly on small problems.
- Integrated graphics allows for fast generation of simple charts and graphs.

Potential Problems with Spreadsheets

Skill levels generally progress from creating a basic spreadsheet (just values, labels, and formulas) to designing spreadsheets making use of 'macros'.

- Spreadsheets are difficult to document.
- Spreadsheets can get very complicated.
- It is easy to make hard-to-find mistakes.
- Spreadsheets operate slowly on large problems.
- Poor initial design can make it very hard to modify and revise the structure of the spreadsheet, as needs change.
- Spreadsheets are often used for applications for which they are inappropriate, in particular as a substitute for data bases.
- It is hard to generalize a spreadsheet to make it useful over a range of similar problems.
- Even basic spreadsheets can be quite complex.

"This [errors in spreadsheets] is a major problem with this technology, as it is very hard to figure out what is going on in the spread sheets, because these formulas are not exactly self-explanatory. There is a whole cottage industry in creating utilities that work with Lotus and other spreadsheets to help you figure out what is going on in the spreadsheet." [Collins]

"Spreadsheets are among the most insidious of programming languages. One proof of this is that most people don't realize that spreadsheets are a programming language. Behind the face of every spreadsheet lie undocumented formulas and incomprehensible macros. These capabilities form a programming language with the same capacity for bugs as languages such as Fortran or Basic. Because spreadsheet programming is hidden behind the face of the spreadsheet, and because there need not be any apparent structure to the spreadsheet, it is difficult to test and debug a spreadsheet.....The important point is that a spreadsheet must be tested and debugged as carefully as any form of computer program, and it may be difficult to do so." [Orenstein]

Because of the possibility of many hidden 'bugs' in spreadsheets, a type of computer program has been developed, called a 'spreadsheet auditor'. This type of program takes an existing spreadsheet and performs a logical analysis and documentation of the spreadsheet, and is very valuable in analyzing complex spreadsheets.

Data Base Management Software

A Data Base Management System (DBMS) is a computer analog of a filing system.

Originally developed for large machines, the technology and concepts have been transported to microcomputers.

Data Base Management Systems (DBMS) allow for storage, retrieval, editing, computation, reports and computations for text and numeric data. Information is generally organized as one or more 'tables', in matrix or 'row and column' format. Each row of the table is usually called a **record**, and the column headings of the table are referred to as **fields**. Programs that handle only a single table at a time are often called 'file managers', and are simpler than programs that handle multiple, related tables (frequently referred to as 'relational data base systems').

Most data base management programs require that field information be specified based on the type of data to be stored (a date, a number, text) and the maximum number of characters that can be stored in the field (the 'field length'). Terminology often varies - different programs refer to the same concepts by different names (e.g. a table may be called a file, database, or relation). dBase III is currently the most popular package for microcomputers.

Data bases are created and used in a five-step process:

- **Design:** Determine the purposes of the database, and define the general nature of the information to be stored (what will be the tables, what will be the records);
- **Structure:** Determine the specific information to be stored, and the manner in which it will be stored (what are the fields, what are the field lengths, what codes will be used);
- **Add Data:** Enter the required information into the data base, either manually (by typing) or through automated transfers from another computer-readable source;
- **Select/Edit/Retrieve/Sort:** Modify the data as necessary, set criteria for the items of data that are to be retrieved from the database, and determine the order that output information is to be presented in;
- **Calculate/Display:** Perform computations (e.g. count the number of items satisfying a specific set of criteria,

The 'table' is the basic organizational structure for microcomputer data bases.

or calculate an average, and present the selected information, in a 'default' or user-specified format, on the computer screen or in printed form;

A user can interact with a DBMS in a variety of manners:

The design of a data base (the definition of the tables and fields to be used) is critical, determining the flexibility, efficiency, and ease of use of the data base application.

- browse through the data base (editor)
- enter commands in 'english-like' language (query language)
- follow a 'menu' of steps to operate the program (assist or prompt mode)
- write special sets of instructions in the program's own internal programming language, to perform tasks not handled by the system's 'standard' capabilities (e.g. complex statistical calculations), or to develop a system that is easy for untrained individuals to use.

In developing an application with a data base management system (DBMS), it is first necessary to develop the data base structure. The tabular form of organization provides the general approach. Definition of the data base structure is equivalent to defining what types of information will be stored in each column of each table. A column is also called a 'field'. A data base structure is generally defined by specifying, for each field (column):

Simple data base applications involve a single table.

More complex applications involve the use of information in multiple tables.

- the type of data to be stored, e.g. a date, a number, a dollar amount, text information, etc.
- the size of data to be stored, e.g. the number of characters to be held in the field
- Additionally, maximum and minimum values, sets of permissible values, and specific formats may be associated with a given field (depending upon the particular DBMS).

The CEHYDRO data base system is an example of a Corps-developed dBase III application. CEHYDRO is a menu-driven system written in the dBase III programming language, and containing information on the status of Corps and non-Federal power facilities and license activities at Corps dams.

"The same set of data may be arranged in many alternative ways within a DBMS. In general, a 'good' design will result in a more efficient data base in terms of usage of space and ac-

Once data is inside a data base, it is presumed to be correct.

It is much harder to find bad data once it is entered, than to keep it from being entered in the first place.

cess time, and in ease of access of the information. Though most DBMS will allow the user to restructure a data base after it has been constructed, it is best to put some 'upfront' effort into data base design to meet the goals of the particular application. There is no single 'best' design for all uses. Design is a function of the intended use of the data base. Therefore, the initial step in designing a data base is a definition of the purpose of the data base and how the data will be used." [Grayman]

"People don't realize the costs of data bases. The major cost of data, as a matter of fact the only significant one, is data entry...there is another aspect people don't consider: after you have got the data in, you have to check the data to see that it is right. Data checking often costs as much as data entry. It may be more expensive because the data entry can be done by fairly low cost people, but data checking frequently needs to be done by someone who knows what the data is supposed to look like." [Collins]

Communications

Uploading:

Moving information from a microcomputer to another computer system, generally over telephone lines.

Downloading:

Moving information from another computer to a microcomputer, again generally over telephone lines.

Electronic Mail:

Using computers connected by communications lines to send and receive messages between individuals or groups.

It is frequently necessary to transfer information between computers. Information available in a large data base on a mainframe computer may be needed for manipulation in a microcomputer application. Conversely, information generated on a microcomputer may need to be transferred to a large centralized system. This transfer is accomplished by 'uploading' and 'downloading'. A telephone link is established between the two computers, and a file is transferred between the computers. Uploading and downloading require special hardware (a modem) and software (a 'communications' program). The modem allows information to be transmitted over phone lines. A modem is required at each computer. Modems vary primarily as to the speed by which they can transmit information. The most common speed for microcomputers is currently 120 characters per second, but higher speeds are becoming increasingly common. A communications program provides the control software for the data transfer. It allows for 'dialing' the distant computer, and performs the translation of information to and from signals sent by the modem. Different forms of translation, called 'protocols', are possible. Compatible protocols must be available on both linked computer systems.

A simple, general-purpose system for 'marriage' of mainframes and microcomputers is not really available at present.

Many large computers are not set up to work easily with microcomputers. Microcomputers can be set up to emulate a terminal for the large computer, and information can be 'captured' by the microcomputer as it appears on the screen, but direct, high-speed transfers require special protocols, not always available on mainframe systems. Uploading/downloading is generally accomplished on a 'batch' basis. That is, an entire file of information is transferred between the two systems at once, with no interactions other than the commands to send and receive the file. More complex interactions are difficult to accomplish. If, for example, an interactive program exists on a mainframe computer, in which the user is prompted with a series of questions, it is possible, but not easy, to store the responses on the microcomputer, and transfer these to the main program.

In general, if information is to be transmitted between a mainframe and microcomputer, the information must be available as 'text' information, i.e. information that can be formatted as printable characters (A-Z, 0-9, etc.). A particular method of representing such text information is known as an 'ASCII' (American Standard Code for Information Interchange) format, and is almost always available on any computer. Thus, the 'ASCII' file is the most universal, transferable format of information between computers.

Microcomputer data storage formats for data bases and spreadsheets are generally incompatible with mainframe computer storage formats.

Communications between compatible microcomputers can be accomplished very easily.

Much uploading/downloading between microcomputers and mainframes has as its source or destination a spreadsheet or database program on the microcomputer. The transmitted information must frequently be reformatted, either on sending, or receiving, or both. This re-formatting may be complex, time-consuming, and may result in loss of information. Mainframe-microcomputer information transfer is thus not always simple or satisfying. Transmission of information between compatible microcomputers, however, can be much more straightforward. Data and programs in a wide variety of formats can be readily transferred, without the necessity for reformatting of information.

ONTYME is the Corps' electronic mail system.

Electronic mail, a related area of involving communications and text processing, allows for transmission of documents and messages electronically between computers, and is an increasingly popular form of communication in large organizations. The Corps uses the ONTYME system as its electronic mail system. Messages can be directed to a single individual or a group, and comments can be added to documents as necessary. An advantage of electronic mail is that you don't end up playing 'telephone tag' - a message is left in the computer, and is available whenever the recipient chooses to look at it.

Programming Skills In End-user Computing

End-user application development is a form of programming.

Development of spreadsheets and databases involves 'programming' decisions. There are even 'programming' decisions involved in much text editing and word processing, and in organizing for communications applications. There are learnable skills associated with design and development of applications, and people tend to get better with experience. A discipline of programming exists, and there are better and worse styles of programming.

End-user computing in spreadsheets and databases can be carried out at a number of levels:

- **data entry**: simple functions, following a set of instructions, requiring little knowledge of the particular program involved; someone else creates the application.
- **command level**: operating in the language of the application, but with a full range of capabilities available, and the ability to create and modify applications
- **programmer level**: operating in the programming language of the application, with the ability to perform tasks beyond the capabilities of the command level user (more complex manipulations, or development of systems that are easy to use at the data entry level).

Development of applications to be used by others requires more skill than developing 'personal' applications.

Each of these levels requires increasing skills and correspondingly more detailed knowledge of the program being used (e.g. the spreadsheet or the data base). **In general, the greatest effort and skill in programming is associated with making an application easy to use at the 'data entry' level.**

A manager must realize that his staff may encompass these varying levels of skill in using a microcomputer. Applications have to be developed with the skill of the end-user in mind

"All programmers are optimists." [Brooks]

"Programming managers have long recognized wide productivity variations between good programmers and poor ones." [Brooks]

"I find quite a lot of people expect to be really good [at programming] after a short time, but I haven't known too many people who have been successful in doing that. Success comes from doing the same thing over and over again; each time you learn a little bit and you do it a little better the next time." [Sachs]

Personnel Issues

Managers need to be aware of the special issues involved in dealing with people as regards computers. There are various 'types' of people when it comes to dealing with microcomputers, with different styles and behaviors:

- *computer-phobes*
- *casual users*
- *regular users*
- *power users*
- *computer 'junkies'*

Managers must understand the way in which individuals 'relate' to computers, as well as technical concepts. There are various types of computer users, needing various degrees of encouragement, support, guidance, and control.

Types of Computer Users

There are a number of different types of individuals who use (or do not wish to use) microcomputers:

- **computer-phobes**: individuals who fear use of computers. They may fear change, technology, be intimidated by supervisors, have unrealistic expectations, fear loss of status if they cannot handle the technology, or fear their loss of control as they move into arenas with which they are not familiar. These individuals need encouragement and reassurance, and opportunities to experiment and learn in a protected setting (one in which their failures or ignorance will not be obvious or significant).

"The crux of most people's resistance to operating computers is fear. After all, these modern-day Houdinis can't help but arouse some uncertainties about your ability to understand and control them." [Kilpatrick]

"Apprehension is a natural response to change. Because technological change requires the learning of new skills and procedures, employees might resist strongly - if they doubt their ability to adapt - or remain unconvinced of the value of the change itself. Many believe that technological advancements are unnecessary evils that lead to a mechanized and impersonal working environment."

"Workers often respond negatively to the frustration of communicating with a system whose inner workings are beyond their comprehension...They also fear equipment malfunction and envision a computer shutdown. Worst of all, new users hear horror stories about screens going blank and worry that a day's work could suddenly and irretrievably disappear."

"With the introduction of new technology and computer automation, veteran staffers might believe that their time at a

company will be instantly erased, and that their years of knowledge will be of little significance. This fear can be compounded if the firm begins to hire new and younger employees with computer training and experience." [Oromaner]

- **casual users**: those who will occasionally use a microcomputer, but do not routinely use computers in their day to day work. Training and support are required, plus opportunities to show these users how they can increase their productive use of computers. Because casual users tend not to grow in their skills, they can be an ongoing burden on the support mechanisms in place.

"Microcomputer users in general are impatient. They require continuous handholding, feeding, and TLC." [Currie]

- **regular users**: computer users who have integrated computer usage into their everyday work styles. Regular users need continued opportunities for training, ongoing support mechanisms, access to libraries of books and programs, and opportunities to share ideas and tips (users groups). Regular users need to be encouraged to conform to organizational norms, and must be managed to insure that they are devoting appropriate resources to the tasks at hand, have instituted adequate backup and documentation mechanisms, and that the applications that they develop fit the organization's needs.

"We can expect that perhaps ten to twenty percent of our potential users will take to personal computers with little or no formal training. An additional thirty to forty percent of the users will require a moderate amount of training to get them started and to help keep up their skills. At the other end of the spectrum, five to ten percent will never go near a personal computer on their own, and will fail miserably if forced. Training is wasted on them. This leaves thirty to fifty-five percent who will require heavy levels of training if they are to succeed in using personal computers." [Orenstein]

- **power users**: the more advanced users, who are most knowledgeable about computers and can serve as a source of support and advice to others. Power users regularly 'track' advances in hardware and software, but may be excessively concentrated in a single area (e.g. spreadsheets). Power users require management control of the applications they develop, which may be excessively personalized. Power users need the opportunity to experiment and learn, at high levels, but must be encouraged to share and teach their knowledge to other levels of users, so that they become a 'corporate' resource. Power users are frequently protective and jealous of their expertise status, at the same time enjoying the advantages it gives them, and often resenting the constant interruptions to assist other users. Managers must give power users clear indications of the expectations that the organization places upon them, and attempt to place them in an organizational framework that formalizes their support and assistance roles.

"There is also some debate whether power users know more about products than how to use them to support the information they need..these users may focus on learning one application to the exclusion of other ones. For example, some users know 1-2-3 so well that they use it for word processing instead of doing their work more efficiently with a word processing package."

"Most power users have to choose between their profession and a technical [computer] career if they want an extended career path. Some users find unofficial positions in their departments in which they combine company business with end-user support...because the position is not officially recognized, upper management may question the time devoted to helping co-workers."[Hurst]

- **computer 'junkies'**: individuals who are strongly interested in computers, to the point of subordination of other aspects of their work. As with power users, computer 'junkies' can be an important resource, but clearly must be managed so that the organization's needs are made primary.

These different types of computer users present different problems to the planning manager. The less advanced users are a management problem, primarily in terms of foregone opportunities [increased productivity through wider use of microcomputers]. The management problem is to create the appropriate environment that encourages and supports more widespread use. The more advanced users, however, can create situations in which significant resources (primarily in terms of time/effort) are expended, often for little 'corporate' gain, and so are a management problem of an entirely different order, requiring monitoring and control.

Behavioral Characteristics of the Computer-Oriented Worker

There are frequently management problems when dealing with computer-oriented individuals:

- *they insist upon interesting work, and will find ways to make work interesting;*
- *they resist routine 'boring' tasks;*
- *it is sometimes hard to communicate with them.*

Problems associated with managing and communicating with computer professionals have been recognized for some time, and various studies specifically related to managing data processing personnel have been carried out, in recognition of the apparent differences between such employees and other employee categories. Some of these findings are of interest to managers

Motivation theory defines two dimensions of needs associated with work satisfaction:

- Social Needs Strength (SNS) - need for, and reinforcement from, interacting with others
- Growth Needs Strength (GNS) - need for learning and developing skills

Studies of data processing personnel from a motivational point of view, as compared with other types of workers, show that they have:

- a lower need for 'social interaction' with co-workers (low SNS)
- a higher need for 'interesting' and 'challenging' work (high GNS)
- a higher need for feedback (either from the work itself, or from the outside)

"One survey revealed two characteristics of computer personnel that require special management action - their low social need and their high growth need. In addition, the scarcity of qualified personnel necessitates special care in managing

computer personnel: they will be more vocal about their feelings because the risk of retribution is low, and they will not be very patient about promised changes....This outcome [high GNS for data processing staff] is no surprise for data processing managers used to demands by their staff that they be provided training, be allowed to attend conferences and seminars, and so on. ... A job low in motivating potential will frustrate a person with high GNS."

"...to be successful, programmers need far less skill in verbal communication. Nor is understanding of behavioral patterns a prerequisite to success in programming."

"What is the typical career path in the systems department? The path is through programming to analysis. So - employees carry their low SNS with them on up the career ladder ... The characteristic of low SNS of DP personnel may be the prime factor in the perpetual difficulty in maintaining satisfactory relations with users of DP."

"People with low social needs can be expected to interact less with subordinates. Communication skill may not come naturally for such people. ..." [Couger]

Implications for Managers

Computer-oriented workers tend to modify their work so that it is interesting to them:

- *they do the parts of the work that are 'fun', such as programming, and avoid the parts that are not fun, such as documentation or reporting*
- *they decide to learn a new program, or new skills, to solve a problem, even if existing techniques would be adequate and appropriate.*

Managers must recognize how computer-oriented personnel may differ from others they manage. The computer-oriented personnel tend to be less able to communicate well (written and spoken); tend to be 'loners', working less effectively in groups; and prefer non-repetitive tasks.

"Some [computer-related] tasks contain a high degree of the five core job dimensions: skill variety, task identity, task significance, autonomy, and feedback. An example is system design. Such tasks are called 'high-scope' tasks. Other tasks contain a relatively low degree of the core job dimensions. An example is system documentation. Such tasks are called 'low-scope' tasks. ... Individuals vary in their need for growth, for challenging work...the goal in motivating personnel is to match the scope of the task with the growth need of the individual". [Couger]

Managers must be aware of the tendencies of more advanced computer users to concentrate on the 'high scope'

tasks, and to re-create the work so that it is interesting to them. Such users will frequently create opportunities to use new computer programs, or to try different techniques, so as to make the job more 'fun' and learn something, even if the particular problem at hand does not require it.

Advanced computer users frequently will avoid tasks associated with writing documentation, organizing information, or developing design documents for applications. These 'low scope' tasks are not as enjoyable, and are thus put off. Managers must clearly and overtly recognize and discuss this problem, and use techniques of job rotation, joint goal-setting, and job enrichment to insure that the essential 'low scope' tasks are accomplished. Joint goal-setting will allow the manager to emphasize the organizational interests at stake (e.g. security, standards, protection against vulnerability) in having the 'low-scope' tasks carried out, and will allow the application developer to make concerns and problems known directly to management. Job rotation can insure that the less interesting tasks are fairly shared, and job enrichment techniques, such as allowing more training and experimentation, can be explicitly introduced as rewards.

Such individuals have a tendency to work alone.

Low social needs strength, the absence of a 'peer' group, and the ability to get feedback from the job itself (programming provides very rapid feedback on success or failure), mean that communications aspects of applications development may be minimized. This can lead to a variety of problems, including inadequate design, poor choice of approach, lack of documentation, and totally 'personalized' applications (designed and built by a single individual, even if the application should properly involve the input of others in the organization).

Skill levels of computer personnel vary significantly. They are not equally competent at various tasks.

Communications tasks and skills (oral and written) are essential in design, construction, documentation, and evaluation of computer applications. In the absence of natural tendencies to communicate, formal training and procedures that insure communication and group effort must be put in place, in particular requirements for written documentation, and use of group meetings to review applications design and development.

Applications development and use involves a variety of skills, which may not all be present in a single individual. Skills include: interviewing for requirements analysis, system design, programming/coding, testing, documentation/technical writing, and data entry. Many computer departments, with sufficient staff, assign these jobs to different individuals. For microcomputer applications development, this may not be possible. Managers must understand the different aspects of the applications development process, and encourage and demand that applications developers improve skills in the arenas in which they are lacking.

Management of 'Corporate' Applications Development

Once the personnel issues are recognized, **the focus of management efforts should be on the application.** An application is (or should be) a finite, concrete product, thus more susceptible to management and quality control. By managing applications, supervisors can control the amount of resource that is devoted to development, and insure that 'corporate' goals and needs are recognized and accounted for.

Goals of Managing Applications Development

- Insure that applications are needed;
- Insure that appropriate resources are directed to development and use of the application so that:
  - the application is available when needed;
  - the level of effort is commensurate with benefits;
  - the build or buy decision is properly made;
  - the right people (with appropriate skills) are developing and implementing the application;
- Insure that good design practices are used;
- Insure that the application is tested adequately before use;
- Insure that the application is easy to modify, when necessary;
- See to it that all relevant parties are included in the design and development of the application;
- Insure appropriate security and vulnerability controls so that the organization is not at risk from:
  - loss of data;
  - incorrect results;
  - transfers of personnel;
  - violations of privacy or security;
- Insure that applications are documented to an appropriate level;
- Insure that applications are reviewed periodically;

Precepts for Management of Applications

- Always develop a 'paper trail' describing the application:
  - requirements analysis (what is needed?)

  - design documents (how will the application be structured?)
- Look at more than one alternative design:
  - consider buying or modifying a ready-made application, as well as building one;
  - Ask other Corps offices if they have developed any similar applications
  - Check electronic bulletin boards (e.g. Corps Planners Board) and printed documentation for possible similar applications
- Perform group reviews of:
  - need for the application
  - design of the application
  - performance of the application
- Insure that more than one individual is familiar with the application.
- Always assign responsibility for the application:
  - key individual
  - backup individual(s)
- Monitor resources devoted to applications development:
  - what is the true cost (including development time, testing, training, and shakedown)?
- Insist on documentation of the application:
  - documentation appropriate to the application's use
  - internal documentation (within the actual programs)
  - external (written, examples)
- Favor modular and incremental approaches to applications development:
  - break the work into reasonable 'chunks' that can show a demonstrated result
- Insist on testing and auditing of the application.
- Do not neglect the associated training and support of the application.
- See to it that you [the manager] understand the application.

For Corps planning managers an important consideration is whether to have planning staff or IM staff develop an application. The answer to that question depends on the relative expertise of the staffs, the availability of the staffs, and the extent of the application. If there are planning personnel who are experienced with the software which is being used to develop the application the advantage of developing the application internally is that the planners will likely know more about the subject of the application than an IM programmer. If the expertise is not available internally, then either the IM staff or an outside contractor would have to be used to develop the application. If the application requires access the other than microcomputer resources, for example, access to an office minicomputer or mainframe, then the IM staff should be invited into the development team. Wherever possible, for corporate applications a development team consisting of planners and IM staff should be assembled to address the application. In this way, the planners can make sure the content of the application is correct, and the IM staff can help to insure that proper procedures are used to develop a solid application.

Applications Development

No application lasts forever.

Each application has a life cycle. It may be short or long. In particular with microcomputers, it is easy to develop short life-cycle applications, such as a spreadsheet designed to be used only once, for a particular purpose. Other applications may have a longer life cycle, but no application lasts forever. Needs change, computer environments change, proponents of the application leave the organization.

An orderly process of development improves any application.

An orderly process should be followed in developing any application. Not all applications require a full-scale, phased development effort, but many do. Any application that will be used by more than one person, or is expected to be used repeatedly, should have, at minimum, a design document, describing the rationale and structure of the application.

Few applications are used only once, and then discarded.

Design can improve any application.

All 'corporate' applications should have a formal design stage.

It is frequently suggested that applications that are only to be used one time do not need to go through a structured development process, nor be documented. **Experience has shown that very few applications really are used only one time, and then completely discarded.** It may be necessary to perform the same task again, based on discoveries of inaccuracy in the original data, or other unforeseen developments. Obviously, there are situations in which a 'quick and dirty' solution, such as a spreadsheet developed for one-time use, or a very simple data base, do not need to go through a complete applications design process. When such work is the basis for other people's work, however, or the use is expected to more extended, the design process should be commensurately more detailed.

"Certainly we write code differently depending on the ultimate use we expect to make of it. But computer centers are full of programs that were written for a short-term use, then were pressed into years of service. Not only pressed, but sometimes hammered and twisted." [Kernigan]

"In the past, the design of many individual programs evolved as the programmer actually coded his assigned functions. The design phase of program development is now recognized to be

a significant factor in determining the reliability and efficiency of the software product. The manpower and time required to implement a given program, as well as life cycle support costs, can be greatly influenced by the design approach and the design alternatives selected." [Barkin]

Phased Application Development

Applications are properly developed through a series of steps or phases, with defined products resulting from the conduct of each phase. In large-scale applications development, checkpoint reviews are conducted before proceeding to the next phase. The typical phases of applications development are:

- Needs Assessment
- Design
- Implementation
- Use and Maintenance
- Decommissioning

NEEDS ASSESSMENT PHASE

Why Do It?

Recognize Need For New/Improved Application:

- Define scope of project
- Outline current operation
- Determine problems and opportunities
- Develop alternative solutions
- Select approach from among alternatives
- Identify costs and benefits
- Recommendations

What needs to be done?

Requirements Analysis:

- Detailed documentation of user's needs
- Definition of inputs, outputs, and processes
- Resource requirements

How will we do it (in general terms)?

DESIGN PHASE

How will we do it (in specific terms)?

- Develop general (functional) design specifications
- Overall design of system
- Detailed definition of products produced
- Design Review/Modifications/Approval
- Detailed specification of system operations

Build it.

Test it.

Document it.

IMPLEMENTATION PHASE

- Programming/Coding/Testing
- Completion of system

- Testing system in "live" environment
- Documentation
- Installation

USE AND MAINTENANCE

Use it
Evaluate and Review
Maintain it

- Ongoing Use
- Performance Review
- Maintenance

DECOMMISSION

Do we still want it?
Archive it?

- Decision/Justification for terminating use
- Archiving - save in retrievable form (don't just throw away)

Basic Types of Applications

Different types of applications will go through different development processes, and require different degrees of design, documentation, and review. While each application must be evaluated on its own, it is possible to provide a rough categorization of basic application types, and associate an appropriate level of design and documentation with each.

Simple Spreadsheet

- a single spreadsheet
  - no macros (spreadsheet programming language)
  - no data import (everything is inserted directly into the spreadsheet)

A project budget can be an example of a simple spreadsheet. Labor hours for different categories are entered, and formulas translate this into dollars based on labor rates. Other expenditure categories are entered, and the total project cost is summed automatically. A manager can adjust labor hours between different categories, and immediately see the effect on the total project budget.

Simple spreadsheets tend to be 'personal' rather than corporate applications.

Simple spreadsheets tend to be 'personalized' rather than 'corporate' applications. As such, they seldom require major organized development efforts. Skilled spreadsheet users can often design directly on the computer for simple applications, or easily modify a previously-developed spreadsheet. Documentation can be limited, and, insofar as possible, should be internal to the spreadsheet. An 'ad hoc' spreadsheet, used for a limited time and a specific purpose, should require only a brief write-up, noting, for the user, the purpose of the spreadsheet, and any particular assumptions that the user has made. This can be stored as internal documentation within the spreadsheet. In any case, the spreadsheet developer should learn and use principles of good spreadsheet design, available in many books and magazine articles. It is good practice to maintain a 'hard copy' example of the spreadsheet output.

More extensive simple spreadsheets can benefit from development of a preliminary spreadsheet design on paper, together with a brief writeup of the use of the spreadsheet, and from application of spreadsheet auditor programs.

"When you're planning a complex worksheet, draw a map showing the layout of the various areas and keep this map updated....A worksheet map also provides basic documentation for anyone else who may later need to modify your worksheets. You might even include the map somewhere in the worksheet itself."

"When building your model, make all your assumptions explicit."

"Document in one area all the assumptions of your model. Often you make certain assumptions in constructing a model. If these assumptions are not clear (to you or to others) when the model is used, confusion can result. It's a good idea to pick a special area where you put information about the assumptions. You can also include special cautions and instructions in the same area."[Andersen]

Simple Data Base

- single table
- design is embodied in definition of fields
- operates exclusively at command level (no programs in the data base programming language)

Encourage applications developers to look at other designs and solutions for similar problems.

An address list, recording first and last names, phone numbers, addresses, city, state and zip codes, is an example of a simple data base. Design decisions relate to the size of fields (e.g. how many characters to store an address), whether to make provision for three or four line addresses, how titles and suffixes (Mr., Jr.) will be stored, etc. Even with such a simple application, there are a number of approaches that can be taken, some better than others; accordingly, some forethought, and examination of the experience and approaches of others solving the same or similar problems, is worthwhile.

Simple databases may only require a single page or two of write-up, serving as design document and documentation of the application itself. A list of the field descriptions for the database (usually easily generated by the database program itself) augmented with descriptions of usage, and a record of the meaning of any codes used within the database, are basic elements of simple database documentation. Use of obvious, descriptive field names is recommended. If special reports or output formats are used, descriptions of these should be recorded, and examples provided.

Complex Spreadsheet

- may involve multiple, related spreadsheets
- makes use of spreadsheet programming language (macros)
- user-defined menus
- data import/export (using or providing data to or/from other programs or computers)

A menu-driven, monthly cost tracking system, with comparison with monthly budgets, calculation of percent complete, and year to date totals, is an example of complex spreadsheet application. Complex spreadsheets may be very difficult to design and document, and are often hard to modify. Careful attention to design on paper before carrying out the application is important. The use of a 'spreadsheet auditor' program that will analyze the spreadsheet for logical errors, and document it, is recommended.

Complex Data Base

- multiple tables
- operation through data base programming language

As data base applications become more complex, it is frequently necessary to make use of multiple, related tables, and to incorporate programming language features, such as customized menus, checking of data entry, and specialized reports. The skill to design and create such applications is greater than that required for simple data bases, and the data base software must have commensurate capabilities. The design of a multi-table data base has significant implications for the ease of use and efficiency of the application, and requires a correspondingly higher degree of attention than does the design of a single table data base. Use of a programming language internal to a

DBMS has many of the same problems associated with applications development using languages such as FORTRAN or BASIC, i.e. much is dependent upon the skill (and style) of the programmer, and problems and difficulties of modification and maintenance may exist, particularly if the application is not well documented.

Programming Language Projects

- make use of custom programming, in a programming language (e.g. Fortran, Cobol, Pascal, C, Basic)

The decision to use a programming language, rather than a data base or spreadsheet, should always be justified and evaluated, not assumed as a given. Programming language projects require significantly more skill, can be inflexible and difficult to maintain, and may be very dependent upon particular hardware and software (i.e. not easily transferable). Use of a programming language is frequently appropriate for scientific and mathematical modeling efforts, but for administrative/managerial work, many of the problems fit more easily into a spreadsheet or data base format.

Combination Applications

- combine the basic types, e.g. custom programming with a data base, or combination of spreadsheet and data base

Combination applications attempt to make use of the best features of commercial end-user software. Developers of such end-user software have recognized the frequent need to import or export information between such programs, and a number of standardized formats have evolved, that are supported by much of the major commercial software. By effectively combining these packages, it is possible to have the advantages of the individual programs, with minimal custom programming. Custom programming can be used to advantage when it is necessary to modify file formats to allow information to pass between programs, or to perform highly specialized operations. Combination applications lend themselves very nicely to organization as independent modules. Developers should be aware however, that future releases of a program that is used in combination with others may not maintain the same data transfer formats, requiring modification to the 'links' between the packages in the combination application.

The above types of applications (complex data bases, programming language projects, and combination applications) have a strong 'programming' component, and tend to be 'corporate' rather than 'personal' applications. As such, they can all benefit from a phased development approach, with at minimum, a design document and design reviews.

Documentation

The purpose of documentation is communication. Complex or unintelligible documentation is no documentation at all.

Managers must be aware of the essential role of documentation in application development. Written documentation of computer applications is important in all phases of the application life cycle:

- during requirements analysis
- during design
- during programming
- in use
- during performance review
- for maintenance and modification of the application
- when the application is archived

Documentation is not just for the purpose of telling someone how to run a program:

- it allows other people to understand (and comment on) the program design
- it can improve the design
- it reminds the programmer how old infrequently used programs function

Documentation is important for 'personal' applications, but it is essential for 'corporate' applications.

Every 'corporate' application, and many 'personal' applications, should be documented at some level. Documentation is necessary for spreadsheets and data bases, as well as programming language applications. **Applications should be documented before and during development, not just after the fact (as is usual).**

Documentation should be developed as the application is developed.

A manager should always be able to see a document describing the application, no matter what stage of development it is in.

"Documentation is used as a reference tool throughout the life cycle of a program or system of programs. It is used during the initial development phase, during the operational phase, and during the maintenance phase. When a program or system of programs is being developed, documentation provides a guide for programmers and designers. ... [Program] specifications are a form of documentation. They tell a designer or programmer what tasks each program should perform and what limitations, restrictions, and considerations must be taken into account by the code." [Spear]

"Documentation is usually done last, or not at all. A good programming practice is to write the documentation before the program is coded. This ensures that the documentation will be completed, and that the planning process will be thorough. Also, the completed documentation becomes a specification for the program. Whether the documentation is done before or after the coding, it will represent a significant fraction of the total programming effort." [Orenstein]

"You can avoid a lot of mistakes when you design programs if you write things down as you go along. First write a problem statement. This is a detailed description of the problem you want to solve....After you've clearly identified the problem, it's time to divide your work into manageable pieces." [Spear]

Internal Documentation

Internal documentation:

- *contained within the 'code' and structure of the application itself, e.g. comments, 'on-line' help, prompts, etc.*

Internal ('in-line') documentation is documentation that is actually embedded within the application itself, and is automatically carried along with it. For a program, this means internal comments written in the program (by the author or program maintainer), prompts that the program itself generates, and 'help' that the program generates. Internal documentation explains what is going on to others who are reading the program, and serves as a reminder to the author of the program, at a later date. Internal documentation must, of course, reflect any changes and modifications that are made to a program over time - that is, as the program is changed, the internal documentation should also be modified to reflect the changes.

"In-line documentation [is] the only kind that seems to be maintained." [Arthur, 1983]

The layout and structure of a program is itself part of the documentation. A good, modular design of a program is more easily understood, and more easily documented, than a complex design.

For a simple spreadsheet, internal documentation consists of text information written within the spreadsheet itself, explaining assumptions, formulas, etc. For a simple data base, internal documentation should consist of reasonably- named data items. Most simple data bases will require additional external documentation. Complex spreadsheets are difficult to document internally, as are complex data bases, and require external documentation. Recently introduced specialized software allows spreadsheet developers to electronically annotate their spreadsheets with notes and comments, similar to yellow stick-on notes placed on a report; such software can significantly improve the limited internal documentation capabilities of the most common spreadsheet software.

External Documentation

External Documentation:

- *documentation that is not 'internal', i.e. documentation that has been generated outside the 'code' of the application (as a separate document or computer file).*

At minimum, a 'program abstract' (a maximum one-page summary) of the program's purpose and functioning should exist. The abstract should provide the author's name, the program purpose, and the hardware, software and data requirements.

Applications that are intended for use by others ('corporate' applications) require more documentation than applications that are thought to be totally 'personalized' (i.e. used only by the developer). It is always worthwhile to assume that eventually, someone else besides the original developer will be working on an application, either as a 'user' or to modify it.

External documentation should exist for almost every application.

"Program users need documentation as a tool to help them successfully run and understand a program....In most cases, users don't care about the nitty gritty details about how the code works. All they want to know is what the program can do for them and what they have to do to make it work correctly. Your users will also look for good error documentation"

"Program users tend to fall into two categories: those who read and those who 'do'...The doers will only turn to a reference booklet as a last resort - when they're hopelessly lost and can't figure a way out by themselves."

"You learn to write as if to someone else because next year you will be 'someone else'" [Kernigan]

"If the program is used solely by the programmer that designed and wrote it, there is still a need for documentation. After writing a dozen programs, you'll find it difficult to remember how to use some of the programs that you run only once in a while. If you need to modify one of your early programs, you'll have to waste time reacquainting yourself with the logic and the code before you can attempt to make changes." [Spear]

Types of Documentation

The type of documentation varies with the phase of application development, and with the intended user of the documentation. Much documentation flows naturally through the applications development process:

- The needs analysis phase generates documentation that serves to define the application, and allows for evaluation of the feasibility of the application. This becomes the basis of the designer's effort.
- The designer produces documents detailing the particular flow of information, and methods to be used in the application.
- This in turn becomes information used in the programming phase. Programmers generate more detailed documentation of the internal workings of the application, useful for their own work, and for maintenance efforts.

Documentation for Needs Analysis

- requirements analysis
- problem statement
- program specifications

Documentation for Designers

- design document
- charts and graphic representations of information flow and program structure (such as flowcharts)

Documentation for Programmers

- 'in-line' documentation (comments in code)
- file descriptions (structure of data files)
- variable lists (names used in the program)
- data dictionaries (characteristics of data items)
- pseudocode (english-like description of program functioning)
- test data

Documentation for Users

- step-by-step instructions

- program overviews
- self-study guides
- tutorials
- reference manual
- 'help' screens
- sample reports
- sample data

Documentation for Maintenance

Documentation for users and managers, not being strictly necessary to make the application work, is often neglected.

Managers must insist that this documentation be developed.

- annotated program listings
- history of changes/modifications
- technical documentation

Documentation for Managers and Potential Users

- program abstract/overview
- sample outputs

Creating Documentation

Documentation can be generated:

- by writing it (preferably on a word processor)
- by computer programs that examine an application, and produce information on how the application is structured (particularly useful for databases and spreadsheets)

Computer-assisted documentation development provides many advantages. All documentation should be created and maintained on word processors, due to the frequent needs for revisions and changes.

Programs exist that will 'document' a spreadsheet, displaying a variety of internal information on the formulas and structure of the spreadsheet. These same programs also provide methods for checking (auditing) the spreadsheet.

A variety of programs also exist that operate upon database programs, producing documents and tables that describe the database structure, and the flow of information in the database programming languages.

So-called 'productivity' software is also of great value in documenting applications. One type of software allows the user to run two programs simultaneously, thus allowing for the ability to make notes about the application while the application is running. Another type of such software allows for any information that is present on the screen, or that would be normally sent to a printer, to be 'captured' in a data file, where it can be edited by a word processor. In this manner, it is very simple to include example reports, computer prompts, etc., in documentation.

Recent developments in the field of 'computer-aided software engineering' (CASE) have also allowed large-machine software design tools to be used on microcomputers. This CASE software is primarily oriented towards assisting designers in developing program designs for large and complex systems. In particular, it can produce a variety of graphical displays of the interrelationships and data flows within an application.

"Most documentation manuals can be divided into sections. These sections might include: introduction, narrative description of program; step-by-step instructions; sample reports; charts; code-listing; appendixes of commands, error messages, and other reference information." [Spear]

"To write step-by-step instructions: sit down at the computer with paper and pencil - write down everything the user must do and everything he will see on the screen. You may use a tape recorder and talk into it. Type up the instructions, number them in sequence, and give them to somebody who doesn't know the program to run them." [Spear]

Insuring Quality in Applications

All corporate applications should have a formal design effort.

One of the most common reasons that applications fail is inadequate effort in the design stage. The existence of a formal design stage, creation of a design document that clearly specifies what the application will do (and how), and group reviews of the design all lead to better applications.

"For some years I have been successfully using the following rule of thumb for scheduling a software task:

- *1/3 planning [design]*
- *1/6 coding*
- *1/4 component test and early system test*
- *1/4 system test, all components in hand" [Brooks]*

Allocate adequate resources to the design effort.

"The cost of the feasibility study should be approximately 5 to 10 percent of the estimated total project cost." [Davis]

Document the design effort.

Managers should demand that a design document exist for any complex, corporate application (one that is expected to be used over time, or developed by one group or individual for use by another). The design document should be intelligible to the manager, and should contain discussion of the need for the application, the proposed approach, the anticipated level of effort, and the probable outputs of the application.

"In many software projects, people begin by holding meetings to debate structure; then they start writing programs. No matter how small the project, however, the manager is wise to begin immediately to formalize at least mini-documents to serve as his data base." [Brooks]

Perform group reviews of the application.

Group reviews have been found to be particularly valuable for improving applications. All too often, an individual who acts simultaneously as needs analyst, designer, and applications developer, can get too close to the application and too committed to the proposed design. In a group review, the designer formally or informally presents the proposed application to a small group of peers, who can comment and provide suggestions. The group review also insures that more than one person is aware of the application's design.

Look for existing applications that solve the problem, or that can be modified.

There is an unfortunate tendency, on the part of many applications developers (particularly those new to computers) to work out a problem on their own, without reference to already existing solutions. Applications tend to fall into categories, with basic types of solutions. Frequently, another group will have solved a similar or identical problem. **Examining the solution approaches of others may obviate the need for development of a new application.** At minimum, it will provide valuable ideas for the design and implementation.

Existing solutions come in a variety of forms and formats. If a commercial data base or spreadsheet is used as the basis of an application, as is common, there are many books that contain 'ready-made' solutions, in the form of data base structures, programs, and spreadsheet templates for a variety of purposes. Often, it is possible to obtain this information in 'computer- readable' form, i.e. on disk. Such 'solutions' can often be readily modified, if they do not provide for the complete set of needs.

In the Corps, given the *similarity* of purposes, many similar applications get developed in different Districts and Divisions. **It is always worthwhile to spend some time looking at what has been done elsewhere.**

Choose the appropriate software tool for the application.

Commercial applications packages (spreadsheets and databases) are often 'the right tool for the job'. It is, however, fairly common to see spreadsheets applied to 'data base' problems, and data bases applied to 'spreadsheet' problems. There is a great tendency to stick with what one knows, and if only a spreadsheet program or data base program is familiar, then that will be applied to any problem, appropriate or not.

In general, applications that involve 'custom programming', i.e. in a programming language such as Basic, Fortran, Pascal, etc., should be avoided. Such programs are more difficult to develop, document, and maintain than those based on commercial packages, and require developers with greater skill and training. Where possible, custom programming should be confined to small modules of specific defined purpose, often to modify data and pass it between different commercial packages.

It is frequently possible and advantageous to develop an application through combination of various commercial packages - for example, a project management package can be combined with a database or spreadsheet package to satisfy the needs of the application. A number of standard 'interchange' formats exist that allow for relatively simple transfer of data from one package to another, and many commercial packages support these formats. Using such an approach, it is frequently possible to minimize the need for custom programming to pass information between packages, if not eliminate it entirely.

Develop applications incrementally and iteratively.

In particular for complex applications, **it is important to structure the development in a modular fashion, with separate components that have defined functions, inputs, and outputs.** A modular approach allows for independent development and testing of each module, and permits the creation of intermediate products, rather than waiting until the entire application is complete before any products can be seen. Simple modules can be replaced at a later date with more complex or powerful modules, if need be. 'All-or-nothing' approaches are more difficult to understand, design, and develop, and are more likely to be error-prone.

It is in the nature of applications to have needs change as the application is developed. As the perception of what can be done becomes apparent, the desires for what should be done will change. This change is normal and should be planned for.

"In most projects, the first system built is barely usable. It may be too slow, too big, awkward to use, or all three. There is no alternative but to start again, smarting but smarter, and build a redesigned version in which these problems are solved. The discard and redesign may be done in one lump, or it may be done piece-by-piece. But all large-system experience shows that it will be done. Where a new system concept or new technology is used, one has to build a system to throw away, for even the best planning is not so omniscient as to get it right the first time. The management question, therefore, is not

whether to build a pilot system and throw it away. You will do that. The only question is whether to plan in advance to build a throwaway, or to promise to deliver the throwaway to customers....hence, plan to throw one away; you will anyhow." [Brooks]

Don't create a number of special cases - generalize the problem and solution.

The ease with which modifications can be made to spreadsheets and databases creates a tendency to develop a variety of applications, for special purposes, that are small variants of a basic 'master' application. **The extra effort needed to create a general-purpose application is frequently worth it**, even if each individual instance does not take advantage of all of the built-in features. When there are multiple, slight variations of an application, it is very difficult to keep track of which version is being used. When modifications are desired, it is not clear which version should be modified. Maintenance and documentation are correspondingly more difficult than if there is a single application.

Things to Do to Improve Applications Quality

A manager can improve the quality of applications by insisting on the use of one or more of the following techniques of tracking and review of applications development.

Development Log

Just as engineering projects should maintain all preliminary design and development work in a single place, it is good practice to maintain a development log in which information pertinent to application design and development is written.

Design Document

At minimum, a one-page justification and explanation of the proposed application should be developed and approved prior to starting any work involving more than a half person-day. The document should contain a rationale for the application, and a discussion of the various design issues and considerations that have been examined.

Group Review

The applications developer should go over the design with at least one person. The design should be reviewed from a managerial point of view (need, resources, schedule, cost/benefit) and from a technical point of view (is the software appropriately chosen?, is the design well carried out?)

Internal Documentation of the Application

The application must be internally documented. Programming language code must contain comments. Spreadsheets must be properly organized, and macros must be commented.

Assignment of Responsible Individual and Backup

One individual, plus a backup, should be responsible for the application. The organization is placed in a vulnerable situation when only a single individual is familiar with an application.

Change Control

Changes in the scope of the application should be documented. Modifications to the application should be clearly identified, and version numbers used to distinguish between revised versions of the application.

External Documentation

For spreadsheets, at minimum a one page write-up should be prepared, together with example output. The writeup should describe the spreadsheet purpose, and any special techniques used.

For simple databases, a one page description of the database, including definition of each field, with sample data and a data base structure, is appropriate. Complex databases require description of the relationships between data, and of any programs used, in addition.

Programming applications require, at minimum, internal documentation, a program abstract describing the overall program structure and purpose, a user's guide, and sample input and output.

Tools for Applications Development

Productivity Tools

Productivity tools are programs that are designed to assist the user in making better use of a microcomputer. As users develop more skills, they need additional capabilities that allow them to keep track of what they are doing, move information around between applications, and make notes.

Over time, skilled microcomputer users will assemble a set of such tools, that greatly enhance their ability to use the computer productively.

Managers should be aware of the capabilities of these productivity tools, in particular for purposes of reducing vulnerability (i.e. disaster recovery tools), and for organizing computer-based information (which can rapidly become disorganized), and should encourage their use. Use of a consistent and common set of productivity tools within a work group is recommended, to allow for interchange of information and skills.

Productivity tools include:

- 'desktop accessories' - programs that provide utilities such as calendars, notepads, and calculators, allowing a user, while working in one application, to get access to these additional capabilities, analogous to having these tools available on a desk. The 'notepad' capability is particularly valuable for design/documentation, in that the applications developer can keep running notes and a log, during development and use of an application.
- file managers and organizers - programs that allow a user to keep track of, and organize, computer files; keeping such files organized can be a major task, particularly where many users are sharing a computer with a hard-disk;
- disaster recovery tools - programs that assist in fixing files that have been damaged or erased;
- backup tools - programs that allow for relatively easy backup and restoration of computer files, in an organized fashion;

- communications software - programs for communication between microcomputers and other computers (mainframe or micro), and fcr transfer of data between computers;
- menu generators - programs that create a simple to use 'user interface' for running programs;
- keyboard and printer utilities - programs that simplify data entry from a keyboard, or allow manipulation of the format and style of output to the printer;
- outliners - specialized forms of word processing programs, that allow for rapid generation and modification of outlines.
- multi-tasking programs - programs that allow two or more other programs to be used simultaneously, such as a spreadsheet and database, with the user able to switch back and forth between them.

Computer Applications Development Aids

Managers should be aware of the great variety of tools that have grown up to support the more popular applications development programs (e.g. dBase, Lotus). In any environment in which these programs are used extensively for applications development, it is worthwhile to explore these tools. Books and disks that contain useful programs, 'libraries' of techniques, and 'templates' (predefined spreadsheets that handle commonly-required tasks, such as calculating rate of return, mortgage payments, etc.) are also very worthwhile in assisting applications developers.

'Add-on' Software

Although commercial applications packages (spreadsheets and databases) can provide a great deal of capability to the application developer, a subsidiary industry has grown up to provide even more capabilities associated with the highly popular applications packages (e.g. dBase III, Lotus) than provided by the developer. These programs, called add-on's or add-in's, provide enhanced capabilities for applications development and use. Such software includes code generators for data base programs (programs that will write programs in the data base programming language, based on simple user input), report generators that allow more complex reports to be developed from spreadsheets or databases, programs that allow for annotation and explanation of spreadsheets, and 'speed-up' programs that make applications run faster. Use of such 'add-on' software can be very effective in simplifying and improving the overall design, development, and functioning of applications.

Auditing Software for Spreadsheets

"A spreadsheet must be tested and debugged as carefully as any form of computer program, and it may be difficult to do so." [Orenstein]

Spreadsheets are notoriously error-prone. It is very easy to make small, hard-to-find mistakes, particularly in complex spreadsheets, or during modification of previously-developed spreadsheets. By the same token, it is hard to determine the correctness of a spreadsheet, except by performing the calculations by hand, and that is no proof of the correctness of the logic of the spreadsheet. To address the problem, a type of software known as spreadsheet auditing software has been developed. This software operates upon existing spreadsheets, looking for internal inconsistencies and logical flaws. In addition, it can provide reports that serve to 'document' the spreadsheet.

Documentation Aids for Data Bases

Just as the 'auditor' program exists for spreadsheets, similar programs exist that will 'document' a data base application. These programs are particularly useful for complex database programs involving use of the data base programming language. The programs can examine an existing data base application, and produce reports that describe the data base structure, and the logical flow and decision structure that is embedded within the application. Such programs are particularly valuable for people who have to maintain or modify applications developed by others.

Structured Design for Applications Development

Structured Design Methods

Structured Design:

a set of tools, techniques, and approaches to applications development, that emphasizes an orderly process and a modular approach

The difficulty of software development for large computer systems has given rise to a discipline of applications development, and a variety of techniques that attempt to formalize the analysis, design, and programming effort. The goal is to produce 'good' applications, i.e. those that are easy to use, that solve the problem at hand, that are internally well-structured and hence easy to modify, if needed.

"Long complex programs were once marvelled at. Now, computer professionals, organizations that depend on computers, and individual users at terminals, all desire programs that are simple to understand and easy to use. They want programs that are easy to maintain when changes are required." [Bohl]

The related approaches of modularity and top-down design have proven worthwhile in development of applications.

Modularity

Modularity:

development of larger applications out of small, nearly independent functional units (modules) that perform one or a limited number of tasks, on defined input and output

Modular development allows programs to be more clearly structured, permits development of re-usable modules that can be parts of many applications, and allows for independent development and testing of modules.

"Most programs are too big to be comprehended as a single chunk. They must be divided into smaller pieces that can be conquered separately. That is the only way to write them reliably; it is the only way to read and understand them... When a program is not broken up into small enough pieces, the larger modules often fail to deliver on these promises. They try to do too much, or too many different things, and hence are difficult to maintain and are too specialized for general use." [Kernigan]

"The ideal module should be [no more than] a page long." [Ratliff]

Top-Down Design

Top Down Design:

Top down design is applications design through a process of step-wise refinement, where modules are defined at increasing levels of specificity as the design progresses

Rather than work out all the details first, the broad approach to the problem is first specified, and higher levels of detail are added as the work progresses. While this is very much analogous to most engineering design approaches in which conceptual design leads to preliminary design which leads to detailed component design, programmers frequently do not work this way, preferring to start immediately with the details, i.e. writing code.

"[Top down design identifies] design as a sequence of refinement steps. One sketches a rough task definition and a rough solution method that achieves the principal result. Then one examines the definition more closely to see how the result differs from what is wanted, and one takes the large steps of the solution and breaks them down into smaller steps ... From this process one identifies modules of solution or of data whose further refinement can proceed independently of other work. The degree of this modularity determines the adaptability and changeability of the program." [Brooks]

Summary

Managers must accept the responsibility of managing microcomputer usage. This means:

- understanding the issues (technical, organizational, personnel) relating to microcomputer usage;
- developing organizational structures, procedures, and attitudes to manage microcomputer use.

General purpose applications packages (data bases and spreadsheets) have allowed for the development of end-user computing, and applications development by a wide variety of individuals, of varying skill levels and commitment to organizational goals.

The appropriate unit of management is the 'application' - the combination of hardware and software resources that is devoted to solving a particular problem. Managers must distinguish between 'corporate' applications (those involving major commitments of time/effort, or development by one individual for others), and 'personalized' applications (use of microcomputers for individual productivity enhancement). The focus of management should be on the 'corporate' applications. Managers can get things under control by managing the development and use of these applications.

By managing the applications development process, managers can insure that organizational needs for consistency, security, and appropriateness of effort, are met.

Managers should start by performing an 'audit' of the applications under their span of control - does the manager know they exist? are they documented? were they designed, or did they 'just happen'? how much organizational resource went into their development? The results of this audit should indicate the degree of problem.

Managers have various tools at their disposal to manage applications development:

- insist that the application be developed incrementally, through a phased development process with check-point reviews;

- insist that the application, and the development process, be documented;
- see to it that applications development is a group effort, involving more than just a single individual.

Managers also must recognize that microcomputer users, at all levels of skill, need encouragement and support. Users need to be encouraged to increase their skills, and to share their knowledge with fellow workers, both formally and informally. Organizational structures need to be put in place so that training and support can be provided easily.

Closure

Microcomputers are a fact of life in organizations. They can provide great advantages in terms of increased productivity, more effective communications, and generation of better information for decision-making. At present, however, we are at an early stage of use of microcomputers, barely six years into widespread use (see the Chronology in the appendix), and at an even earlier stage of understanding about how to most effectively use microcomputers within the goals and objectives of the organization. While we are still learning, managers must accept for themselves the responsibility of seeing to it that the human, hardware/software, and time resources associated with microcomputers are used effectively.

Management of microcomputer usage is not a simple task, and it will not happen by itself. There is a great deal of technical information, often unfamiliar, that must be assimilated; in addition, the way that people operate with and relate to computer usage needs to be understood. Moreover, microcomputer technology is always in a state of rapid change, only occasionally characterized by periods of relative stability of the technology. Dealing with this environment of rapid change requires an ongoing, not a one-time, effort. Managers must accept the need to learn and understand the nature of this technology.

A manager must see to it that microcomputer usage is encouraged, supported, and controlled. Encouragement means providing a favorable environment, allowing time for experimentation and recognizing the long learning curve that exists, and the fears and uncertainties of some people. Users must be encouraged to increase their skills and capabilities over time. Support means providing the necessary technical tools, human resources, and training that will allow people to use microcomputers productively without having to devote all of their time to it. Individuals are much more ready to try something when they have a 'safety net' beneath them, in the form of a microcomputer support system - generally, an individual or individuals who can 'fix it when it breaks', and get less sophisticated users over the more difficult, technical parts. Control means insuring that microcomputers are used effectively and efficiently, so that the organization is not vulnerable to incorrect information, poorly designed applications, undocumented applications, or excessive reliance on specific individuals. The suggested method of control is oriented around orderly, phased development of microcomputer applications, with emphasis on documentation of each step in the process.

The Corps manager should look to IMO to fill the basic support role, in particular for 'high-level' technical input and expertise. Nonetheless, it is ultimately the manager's responsibility to insure that the end-user computing that is done within his/her span of control is done appropriately, effectively, and efficiently.

Appendix I - References

Andersen, Dick, and Cobb, Douglas Ford "1-2-3: Tips, Tricks, and Traps" Que Corporation, Indianapolis, Indiana ISBN 0-88022-110-0, 1985

Barkin, Shelle, "HEC Software Development Guidelines for Generalized Computer Programs - Draft", US Army Corps of Engineers Hydrologic Engineering Center, Davis, CA

Bohl, Marilyn, "Tools for Structured Design", Science Research Associates, Inc., 1978, ISBN 0-574-21170-5

Brooks, Jr., F.P., "The Mythical Man-Month", Addison-Wesley, 1975

Collins, Bert K., "Productivity Tools: Spreadsheets and Data Base Management", in "Microcomputers in Engineering Practice, Computer Group Lecture Series, February/March 1986", Boston Society of Civil Engineers, Boston, MA

Couger, J. Daniel and Zawacki, Robert A., "Motivating and Managing Computer Personnel", John Wiley & Sons, NY 1980, ISBN 0-471-08485-9

Currie, Edward H., "Users Will Try Anything", Business Computer Systems, July 1983

Davis, William S., "Tools and Techniques for Structured Design", Addison-Wesley, 1983

Frank, Michael R., "The Effective EDP Manager", AMACOM, division of American Management Association, New York, 1980, ISBN 0-8144-5635-9

Grayman, Walter M., "Surface Water Screening Model, A Case Study for Water Utility Management", AWWA Research Foundation, Denver, CO, 1986

Hurst, Rebecca, "Help for the PC junkie", Computerworld, August 12, 1987

Kernighan, Brian W., and Plauger, P.J., "The Elements of Programming Style", 2nd Ed., McGraw Hill, 1978

Kilpatrick, Michael, "Computer Phobia", Infosystems, December 1984

Orenstein, Glen S., "Managing Microcomputers in a Civil Engineering Firm", in "Microcomputers in Engineering Practice, Computer Group Lecture Series, February/March 1986", Boston Society of Civil Engineers, Boston, MA

Oromaner, Daniel S., "Overcoming Staff Resistance to Technological Innovation", Office Systems '86, February 1986

Sachs, Jonathan (author of Lotus 1-2-3), quoted in "Programmers at Work - 1st Series", Microsoft Press, Redmond, Washington, 1986

Spear, Barbara, "How to Document Your Software", Tab Books, Inc., Blue Ridge Summit, PA, 1984, ISBN 0-8306-0724-2

Ratliff, C. Wayne (author of dBase II), quoted in "Programmers at Work - 1st Series ", Microsoft Press, Redmond, Washington, 1986

Appendix II - Chronology of Automated Computation

3000 BC	Abacus Developed
1617	Napier's Bones (forerunner of sliderule)
1630	Sliderule invented
1642	Pascal's Mechanical Adder
1674	Leibniz' Mechanical Calculator
1679	Leibniz develops binary system
1801	Jacquard Loom - programmed by punched cards
1823	Charles Babbage designs Difference Engine
1834	Babbage starts designing the Analytical Engine
1890	Herman Hollerith's punched cards used in 1890 census
1892	Burroughs develops first commercial adding machine
1924	IBM formed
1937	Alan Turing's paper describes a programmed computer
1944	Mark I relay-driven computer developed at Harvard
1945	VonNeumann develops stored program concept
1943-46	ENIAC - first electronic computer
1947	Transistor invented
1949-52	Ferrite core computer memory developed
1951	Remington Rand Univac - first commercial electronic stored program computer
1957	Integrated circuit invented
	FORTRAN computer language introduced
1963	Digital Equipment Corporation PDP-8 (first minicomputer)
1964	IBM introduces the 360 Series computer line
	Dartmouth BASIC computer language
1969	Intel 4004 Microprocessor
1971	Mass-produced pocket calculators
1972	Pong video game developed
1973	CP/M - first disk operating system for micros
	Winchester (hard) disk developed by IBM
1974	Intel develops 8080 microprocessor
1975	Altair microcomputer (first 'personal' computer)
	Microsoft develops BASIC language for Altair
	First retail computer store
	Ethernet local area network for large computers
1976	Apple I
1977	Apple II
	TRS-80
1978	Disk Drives for Apple II
1979	Visicalc spreadsheet program introduced
	Wordstar word processing program introduced
	Early version of dBase II marketed
1981	Osborne introduces a 'portable' microcomputer
	IBM introduces the PC
	Microsoft MS-DOS operating system for PC
1982	Apple Lisa
1983	Lotus 1-2-3 (January)
	IBM PC/XT with hard disk
1984	Apple Macintosh
	dBase III
	IBM PC/AT based on 80286 chip
1987	IBM Personal System/2

END
DATE
FILMED
DTIC
JULY 88